Which face is fake?
Supervised vs unsupervised learning

**Supervised Learning**

**Data:** \((x, y)\)
- \(x\) is data, \(y\) is label

**Goal:** Learn function to map 
\[ x \rightarrow y \]

**Examples:** Classification, regression, object detection, semantic segmentation, etc.

---

**Unsupervised Learning**

**Data:** \(x\)
- \(x\) is data, no labels!

**Goal:** Learn some hidden or underlying structure of the data

**Examples:** Clustering, feature or dimensionality reduction, etc.
Supervised vs unsupervised learning

**Supervised Learning**

**Data:** \((x, y)\)

- \(x\) is data, \(y\) is label

**Goal:** Learn function to map \(x \rightarrow y\)

**Examples:** Classification, regression, object detection, semantic segmentation, etc.

**Unsupervised Learning**

**Data:** \(x\)

- \(x\) is data, no labels!

**Goal:** Learn some *hidden* or *underlying structure* of the data

**Examples:** Clustering, feature or dimensionality reduction, etc.
Generative modeling

**Goal:** Take as input training samples from some distribution and learn a model that represents that distribution.

Density Estimation

Sample Generation

How can we learn $P_{model}(x)$ similar to $P_{data}(x)$?
Why generative models? Debiasing

Capable of uncovering *underlying latent variables* in a dataset

Homogeneous skin color, pose VS Diverse skin color, pose, illumination

How can we use latent distributions to create fair and representative datasets?
Why generative models? Outlier detection

- **Problem:** How can we detect when we encounter something new or rare?
- **Strategy:** Leverage generative models, detect outliers in the distribution
- **Use outliers during training to improve even more!**

95% of Driving Data:
(1) sunny, (2) highway, (3) straight road

Detect outliers to avoid unpredictable behavior when training

- Edge Cases
- Harsh Weather
- Pedestrians
Latent variable models

Autoencoders and Variational Autoencoders (VAEs)

Generative Adversarial Networks (GANs)
What is a latent variable?

Myth of the Cave
What is a latent variable?

Can we learn the true explanatory factors, e.g. latent variables, from only observed data?
Autoencoders
Autoencoders: background

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data.

“Encoder” learns mapping from the data, $x$, to a low-dimensional latent space, $z$.

Why do we care about a low-dimensional $z$?
Autoencoders: background

How can we learn this latent space?
Train the model to use these features to **reconstruct the original data**

“Decoder” learns mapping back from latent, \( z \), to a reconstructed observation, \( \hat{x} \)
Autoencoders: background

How can we learn this latent space?
Train the model to use these features to **reconstruct the original data**

\[ \mathcal{L}(x, \hat{x}) = ||x - \hat{x}||^2 \]

Loss function doesn’t use any labels!
Autoencoders: background

How can we learn this latent space?
Train the model to use these features to **reconstruct the original data**.

\[
\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2
\]

Loss function doesn’t use any labels!
Dimensionality of latent space \(\rightarrow\) reconstruction quality

Autoencoding is a form of compression!
Smaller latent space will force a larger training bottleneck
Autoencoders for representation learning

**Bottleneck hidden layer** forces network to learn a compressed latent representation

**Reconstruction loss** forces the latent representation to capture (or encode) as much “information” about the data as possible

**Autoencoding = Automatically encoding data**
Variational Autoencoders (VAEs)
VAEs: key difference with traditional autoencoder
VAEs: key difference with traditional autoencoder
VAEs: key difference with traditional autoencoder

Variational autoencoders are a probabilistic twist on autoencoders!
Sample from the mean and standard dev. to compute latent sample
VAE optimization

Encoder computes: $p_\phi(z|x)$

Decoder computes: $q_\theta(x|z)$
VAE optimization

Encoder computes: $p_{\phi}(z|x)$
Decoder computes: $q_{\theta}(x|z)$

$L(\phi, \theta) = \text{(reconstruction loss)} + \text{(regularization term)}$
VAE optimization

Encoder computes: $p_{\phi}(z|x)$

Decoder computes: $q_{\theta}(x|z)$

$L(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$
VAE optimization

Encoder computes: $p_{\phi}(z|x)$

Decoder computes: $q_{\theta}(x|z)$

$$
\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}
$$

e.g. $||x - \hat{x}||^2$
VAE optimization

Encoder computes: $p_\phi(z|x)$

Decoder computes: $q_\theta(x|z)$

$L(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$
Priors on the latent distribution

\[ D \left( p_\Phi(z|x) \parallel p(z) \right) \]

Inferred latent distribution

Fixed prior on latent distribution

Common choice of prior:

\[ p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1) \]

- Encourages encodings to distribute encodings evenly around the center of the latent space
- Penalize the network when it tries to “cheat” by clustering points in specific regions (ie. memorizing the data)
Priors on the latent distribution

\[
D \left( p_\Phi (z|x) \parallel p(z) \right) \\
= -\frac{1}{2} \sum_{j=0}^{k-1} \left( \sigma_j + \mu_j^2 - 1 - \log \sigma_j \right)
\]

Common choice of prior:

\[p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)\]

• Encourages encodings to distribute encodings evenly around the center of the latent space
• Penalize the network when it tries to “cheat” by clustering points in specific regions (ie. memorizing the data)
VAEs computation graph

Encoder computes: $p_\phi(z|x)$

Decoder computes: $q_\theta(x|z)$

$L(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$
VAEs computation graph

Problem: We cannot backpropagate gradients through sampling layers!

\[ \mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)} \]
Reparametrizing the sampling layer

Key Idea:

Consider the sampled latent vector as a sum of

• a fixed $\mu$ vector,
• and fixed $\sigma$ vector, scaled by random constants drawn from the prior distribution

$\Rightarrow z = \mu + \sigma \circ \varepsilon$

where $\varepsilon \sim \mathcal{N}(0, 1)$
Reparameterizing the sampling layer
Reparametrizing the sampling layer

Deterministic node

Stochastic node

Original form

Reparametrized form

$z \sim p_\phi(z|x)$

$z = g(\phi, x, \varepsilon)$

$\sim \mathcal{N}(0,1)$
VAEs: Latent perturbation

Slowly increase or decrease a **single latent variable**
Keep all other variables fixed

Different dimensions of $z$ encodes **different interpretable latent features**
Ideally, we want latent variables that are uncorrelated with each other. Enforce diagonal prior on the latent variables to encourage independence. Disentanglement.
VAEs: Latent perturbation
VAEs: Latent perturbation
VAE summary

1. Compress representation of world to something we can use to learn
VAE summary

1. Compress representation of world to something we can use to learn
2. Reconstruction allows for unsupervised learning (no labels!)
VAE summary

1. Compress representation of world to something we can use to learn
2. Reconstruction allows for unsupervised learning (no labels!)
3. Reparameterization trick to train end-to-end
VAE summary

1. Compress representation of world to something we can use to learn
2. Reconstruction allows for unsupervised learning (no labels!)
3. Reparameterization trick to train end-to-end
4. Interpret hidden latent variables using perturbation
VAE summary

1. Compress representation of world to something we can use to learn
2. Reconstruction allows for unsupervised learning (no labels!)
3. Reparameterization trick to train end-to-end
4. Interpret hidden latent variables using perturbation
5. Generating new examples
Generative Adversarial Networks (GANs)
What if we just want to sample?

**Idea:** don’t explicitly model density, and instead just sample to generate new instances.

**Problem:** want to sample from complex distribution – can’t do this directly!

**Solution:** sample from something simple (noise), learn a transformation to the training distribution.
Generative Adversarial Networks (GANs) are a way to make a generative model by having two neural networks compete with each other.

The discriminator tries to identify real data from fakes created by the generator.

The generator turns noise into an imitation of the data to try to trick the discriminator.
Intuition behind GANs

**Generator** starts from noise to try to create an imitation of the data.
Intuition behind GANs

**Discriminator** looks at both real data and fake data created by the generator.
Intuition behind GANs

**Discriminator** looks at both real data and fake data created by the generator.

- **Discriminator**
- **Generator**

Real data    Fake data
Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

\[ P(\text{real}) = 1 \]

Real data 🟢 Fake data 🟥

---

6.8191 Introduction to Deep Learning
introtodeeplearning.com
Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

\[ P(\text{real}) = 1 \]

Real data  
Fake data
Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

\[ P(\text{real}) = 1 \]

Real data

Fake data
Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

- **Discriminator**
  - \( P(\text{real}) = 1 \)
  - Real data: green circles
  - Fake data: pink circles

- **Generator**
Intuition behind GANs

**Generator** tries to improve its imitation of the data.

- **Discriminator**: $P(\text{real}) = 1$
- **Generator**:

\[\begin{align*}
\text{Real data} & \quad \text{Fake data}
\end{align*}\]
Intuition behind GANs

**Generator** tries to improve its imitation of the data.

Discriminator

\[ P(\text{real}) = 1 \]

Real data  Fake data
Intuition behind GANs

**Generator** tries to improve its imitation of the data.

- **Discriminator**
  - $P(\text{real}) = 1$

- **Generator**

![Real data](green-circle.png) ![Fake data](red-circle.png)
Intuition behind GANs

**Discriminator** tries to predict what’s real and what’s fake.

Discriminator

\[ P(\text{real}) = 1 \]

Real data  Fake data
Intuition behind GANs

**Discriminator** tries to predict what’s real and what’s fake.

Discriminator

\[ P(\text{real}) = 1 \]

Real data  Fake data
Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

- **Discriminator**
  - $P(\text{real}) = 1$

- **Generator**

Real data  Fake data
Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

\[ P(\text{real}) = 1 \]

Real data 🌱 Fake data 🍓
Intuition behind GANs

**Generator** tries to improve its imitation of the data.

\[ P(\text{real}) = 1 \]

Real data  
Fake data
Intuition behind GANs

**Generator** tries to improve its imitation of the data.

\[ P(\text{real}) = 1 \]

Real data  Fake data
Intuition behind GANs

**Generator** tries to improve its imitation of the data.

$p(real) = 1$

Real data  Fake data
Intuition behind GANs

**Discriminator** tries to identify real data from fakes created by the generator. **Generator** tries to create imitations of data to trick the discriminator.

\[
P(\text{real}) = 1
\]
Training GANs

**Discriminator** tries to identify real data from fakes created by the generator. **Generator** tries to create imitations of data to trick the discriminator.

Train GAN jointly via minimax game:

\[
\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log \left( 1 - D_{\theta_d}(G_{\theta_g}(z)) \right) \right]
\]

**Discriminator** wants to maximize objective s.t. \( D(x) \) close to 1, \( D(G(z)) \) close to 0. **Generator** wants to minimize objective s.t. \( D(G(z)) \) close to 1.
Why GANs?

A. Courville, 6S191 2018.
Why GANs?

more traditional max-likelihood approach

GAN

A. Courville, 6S191 2018.
Generating new data with GANs

After training, use generator network to create new data that's never been seen before.
GANs: Recent Advances
Progressive growing of GANs (NVIDIA)

Karras et al., ICLR 2018.
Progressive growing of GANs: results

Karras et al., ICLR 2018.
Style-based generator: results

Karras et al., Arxiv 2018.
Style-based transfer: results

Karras et al., Arxiv 2018.
CycleGAN: domain transformation

CycleGAN learns transformations across domains with unpaired data.

\[ DX \quad G \quad DY \]

\[ X \quad F \quad Y \]

Zhu et al., ICCV 2017.
Deep Generative Modeling: Summary

**Autoencoders and Variational Autoencoders (VAEs)**

Learn lower-dimensional latent space and **sample** to generate input reconstructions

**Generative Adversarial Networks (GANs)**

Competing **generator** and **discriminator** networks
References:
https://goo.gl/ZuBkGx9